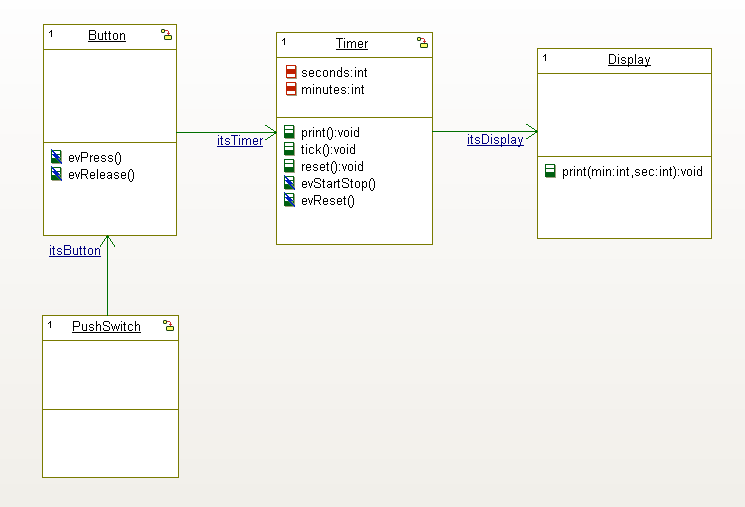
|  |  |
| --- | --- |
| 년도-학기 | 2021년 1학기 |
| 과목명 | 임베디드시스템설계 |

|  |  |
| --- | --- |
| **LAB번호** | **제목** |
| 8 | Rhapsody with Raspberrypi |

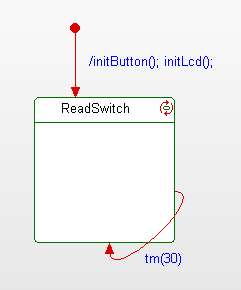
|  |  |
| --- | --- |
| 실험 일자 | 2021년 4월 30일 |
| 제출자 이름 | 강\*\* |
| 제출자 학번 | 201803\*\*\*\* |
| 팀원 이름 |  |
|  |  |

**Chapter 1. 프로그램의 동작 방식 설명**

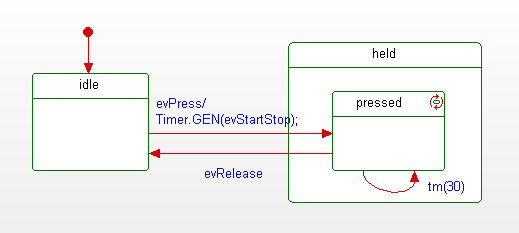
프로그램의 동작 방식을 설명할 때 object model diagram을 참고하면 좋을 것 같아 사진을 첨부했다. 프로그램의 전체적인 구조를 파악할 수 있다. 총 4개의 object를 사용했으며 interface.cpp 파일을 추가해 일부 함수를 구현했다.



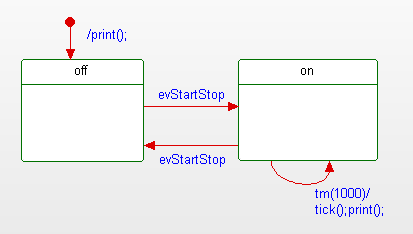
PushSwitch object는 먼저 button과 lcd를 초기화 시킨다. Driver를 열고, lcd에 00:00을 출력하게 한다. 30ms마다 버튼의 입력 값을 확인을 반복하며 버튼이 눌리면 evPress를 발생시킨다.



Button object는 다음과 같은 statechart를 가진다. 처음엔 idle 상태에서 기다린다. evPress가 발생하면 Timer에 evStartStop을 발생시키고 pressed state에서 30ms마다 버튼의 입력 값을 확인한다. 버튼 입력이 감지되면 evRelease를 발생시킨다. 이는 polling 방식에서 버튼 채터링을 방지하기 위함이다.



Timer object에서는 시간을 측정하고 출력한다. 처음 시작하면 콘솔창에는 0:0, lcd에는 00:00이 출력된다. evStartStop이 발생하면 1초마다 tick 함수와 print함수를 실행한다. tick 함수는 시간을 계산하고 lcd에 출력하는 함수를 호출한다. print함수는 콘솔창에서 시간을 출력한다. on state에서 시간을 계속 측정하다가 버튼이 또 눌리게 되면 evStartStop이 발생하고, off state로 가서 멈춘다.



Display object는 Timer의 tick 함수에서 측정한 시간을 콘솔창에 출력하는 함수를 가진다.

위의 기능들을 수행하면서 stopwatch로 동작한다.

**Chapter 2. 결과**



버튼을 눌러 stopwatch를 멈췄을 때 라즈베리파이의 콘솔 창과 lcd가 동일한 시간을 출력함을 보였다. 콘솔창을 보면 프로그램이 종료된 것이 아니라 멈춰 커서가 깜빡이는 상태라는 것을 알 수 있다. 버튼을 누르면 다시 시간이 측정된다.

**Chapter 3. 결론 및 Discussion**

exercise 5에서 stopwatch project를 수정해 lcd와 button을 이용하라고 하였다. 이에 따라 기존 exercise에서 했던 stopwatch project와 button project를 참고해 수정했더니 잘 해결할 수 있었다. 버튼을 잘 누르면 제대로 인식되었고 채터링 문제도 없었다. 버튼을 누르면 시간이 카운팅이 시작되며 1초씩 증가되었고, 버튼을 다시 누르면 멈추는 스탑워치의 기능을 제대로 구현할 수 있었다.

이번 과제는 stopwath를 구현하는 것보다 rhapsody를 이용하는 것이 어려웠던 것 같다. 처음 써보는 프로그램이라 오류가 생겼을 때 어디에 가서 수정할 지 고민하게 되었다. Stopwatch의 구현은 성공적으로 마무리했으나 다만 조금 아쉽고, 원인을 제대로 파악하지 못한 부분이 있었다. 교수님께서 올려주신 Led를 켜는 button project를 실행했을 때는 버튼을 빨리 눌러도 인식이 잘 되었던 것 같은데, 과제도 그것과 같은 방식으로 구현했지만 버튼을 정직하게 제대로 딸깍 눌러야 인식이 잘 되는 것 같다. 문제로 삼기엔 잘 동작할 때가 많고 테스트할 때 가끔가다 인식이 안될 때가 생겼다. 버튼과 관련해선 같다고 해도 무방한데 이유가 뭔 지는 잘 모르겠다. Timer의 tm(1000)과 Button과 Pushswitch의 tm(30)반복 사이에 시간이 맞지 않을 때가 있어 가끔 인식이 되지 않는 것일지도 모른다는 생각이 들었다.

**부록**

// Interface.cpp 코드 첨부합니다. 프로젝트의 파일과 Lcd1602 driver 코드를 별도의 파일로 첨부하였습니다.

#include <fcntl.h>

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <sys/types.h>

#include <sys/stat.h>

#include <string.h>

#include <sys/ioctl.h>

#define IOCTL\_MAGIC 'G'

#define CLEAR \_IOW(IOCTL\_MAGIC, 1 ,int)

#define FIRST\_ROW \_IOW(IOCTL\_MAGIC, 2 ,int)

int fd\_button;

int fd\_lcd;

int initButton(void)

{

fd\_button=open("/dev/gpio\_button",O\_RDWR);

if (fd\_button < 0) {

printf("Device open error : %s\n","/dev/gpio\_button");

exit(1);

}

}

int initLcd(void)

{

fd\_lcd=open("/dev/lcd1602",O\_RDWR);

if (fd\_lcd < 0) {

printf("Device open error : %s\n","/dev/lcd1602");

exit(1);

}

ioctl(fd\_lcd,CLEAR,0x01);

ioctl(fd\_lcd,FIRST\_ROW,0x80);

char wbuf[30];

strcpy(wbuf,"00:00");

write(fd\_lcd,wbuf,strlen(wbuf));

}

unsigned char readButton(void)

{

char buf[30];

read(fd\_button, buf, 1);

return buf[0];

}

void displaylcd(int min, int sec)

{

char wbuf[30];

wbuf[0]=min/10+0x30;

wbuf[1]=min%10+0x30;

wbuf[2]=':';

wbuf[3]=sec/10+0x30;

wbuf[4]=sec%10+0x30;

for(int i=5;i<15;i++) wbuf[i]=' ';

wbuf[15]=0x0;

write(fd\_lcd,wbuf,strlen(wbuf));

}